Electronic Version
Stylesheet Version v1.1.1

Description

[First response computer virus blocking]

CROSS REFERENCE TO RELATED APPLICATIONS

[0001]

BACKGROUND OF INVENTION

[0002] Electronic/computer data viruses represent a potentially serious liability to all electronic data users and especially to those who regularly transfer data between computers. Computer viruses were first identified in the 1980's, and up until the mid-1990s consisted of a piece of executable code which attached itself to a bona fide computer program. At that time, a virus typically inserted a JUMP instruction into the start of the program which, when the program was executed, caused a jump to occur to the "active" part of the virus. In many cases, the viruses were inert and activation of a virus merely resulted in its being spread to other bona fide programs. In other cases however, activation of a virus could cause malfunctioning of the computer running the program including, in extreme cases, the crashing of the computer and the loss of data.

[0003] Computer software intended to detect (and in some cases disinfect) infected programs has in general relied as a first step upon identifying those data files which contain executable code, e.g. .exe, .com, .bat. Once identified, these files are searched (or parsed) for certain signatures which are associated with known viruses. The producers of anti-virus software maintain up to date records of such signatures which may be, for example, checksums.

[0004] WO95/12162 describes a virus protection system in which executable data files about to be executed are passed from user computers of a computer network to a central server for virus checking. Checking involves parsing the files for signatures of known viruses as well as for signatures of files known to be clean (or uninfected).

[0005] US6577920 describes a virus protection system in which data files are scanned to determine if they contain macro code which matches the hash signature of known macro viruses. This does not take into account the complete hash signature or checksum of larger files or executable applications.

[0006] There are a number of problems with these more or less conventional approaches. There is inevitably a time lag between a virus being released and identified and the development and release of an updated virus definitions file. By this time many computers may have been infected. Secondly, end users may be slow in updating their systems with the latest virus definitions. Again, this leaves a large window of opportunity for systems to become infected.

[0007] WO 98/14872 describes an anti-virus system which uses a database of known virus signatures as described above, but which additionally seeks to detect unknown viruses based upon expected virus properties. However, given the ingenuity of virus producers, such a system is unlikely to be completely effective against unusual and exotic new viruses.

[0008] US6577920 describes an anti-virus system which uses multiple databases to determine a hash specific to a macro virus such as those found in Microsoft Office documents that contain macros. The problem with this approach, while effective for some viruses, is that it limits the scope of using checksums for all other types of infected or malicious files.

[0009] The other problem unchanged by US6577920 and WO 98/14872 is the multiple hours to days that are spent while anti-virus companies develop, test and release virus definition files for virus scanning software. This time lag can be crippling for Government agencies, corporations or individuals who would prefer to have capability in place to prevent becoming infected in the first place. They all require a much more effective and much faster means to prevent viruses and other malicious software from harming their networks, servers, computers and other electronic devices.

SUMMARY OF INVENTION

[0010] The first object of the present invention is to overcome or at least mitigate the above noted disadvantages of existing anti-virus software.

[0011] The second object of the present invention is to block, quarantine, delete and/or perform additional actions on viruses or other malicious files using new methods and apparatus.

[0012] According to a first aspect of the present invention there is provided a method of screening a software file for viral infection, the method comprising;

[0013] defining a database of signatures of files that are known to contain a virus.

[0014] scanning said file to determine whether or not the file has a signature corresponding to one of the signatures contained in said database.

[0015] The present invention has the significant advantage that it may be used to effectively block the transfer and/or processing of files which contain an identified virus. It is therefore less critical for virus definition files and other software fixes to be updated immediately or for operating systems to be frequently patched to undo damage that has been done.

[0016] Preferably, said step of defining a database of signatures of files known to contain a virus or otherwise infected file will be portable enough to be executed quickly even on machines that traditionally would have taken considerable time to scan for said infected files in more conventional ways. More preferably, the step of defining the database comprises the further steps of updating the database with additional signatures. This updating may be done via an electronic link between a computer hosting the database (where the scanning of the file is performed) and a remote central computer. Alternatively, the database may be updated by way of data stored on an electronic storage medium such as a floppy disk, CD, DVD, flash device or other peripheral storage device.

[0017] According to a second aspect of the present invention there is provided a method of screening a software file for viral infection, the method comprising:

[0018] defining a first database of known macro virus signatures determining a signature for the file and screening that signature against the signatures contained in said databases; and

[0019] alerting a user in the event that the file has a signature corresponding to a signature contained in said database.

[0020] According to a third aspect of the present invention there is provided an apparatus for screening a software file for viral infection, the apparatus comprising;

[0021] a memory storing a set of signatures of files previously identified as containing a virus; and

[0022] a data processor arranged to scan said file to determine whether or not the file contains a matching hash.

[0023] According to a third aspect of the present invention there is provided a computer memory encoded with executable instructions representing a computer program for causing a computer system to:

[0024] maintain a database of signatures of files previously identified as being infected; and

[0025] scan data files to determine a hash signature; and

[0026] determine whether or not the file has a signature corresponding to one of the signatures contained in said database.

[0027] Preferably, the computer program provides for the updating of said database with additional file signatures. More preferably, the computer program provides a mechanism for quarantine of infected files until such a time as an updated virus definition file can be received by anti-virus software to eradicate or repair said quarantined file before any damage could be done to the users computer or data.

[0028] According to a fourth aspect of the present invention there is provided apparatus for determining and screening partial file hash signatures of files in transit or in situations where only a partial file is visible from a given device, the apparatus comprising;

[0029] a memory storing a set of signatures of partial file(s) previously identified as containing a virus; and

[0030] a data processor arranged to scan said partial file(s) to determine whether or not the file(s) contains a matching hash.

[0031] According to a third aspect of the present invention there is provided a computer memory encoded with executable instructions representing a computer program for causing a computer system to:

[0032] maintain a database of signatures of partial files previously identified as being infected; and

[0033] scan partial data files to determine a hash signature; and

[0034] determine whether or not the partial file has a signature corresponding to one of the signatures contained in said database.

[0035]

Brief Description of Drawings

[0036] FIG. 1 is a functional block diagram of the method of computing a file hash signature and comparing it to a database of known file signatures; and

[0037] FIG. 2 is a functional block diagram of a computer system in which is installed virus blocking software; and

[0038] FIG. 3 is a flow chart illustrating the method of operation of the system of FIG. 2; and

[0039] FIG. 4 is a functional block diagram of the method of computing a file hash signature and comparing it to a database of known file signatures when the file is in transit and is broken into several data streams.

Detailed Description

[0040] For the purpose of illustration, the following example is described with reference to the Apple Macintosh OS X.TM. series of operating systems, although it will be appreciated that the invention is also applicable to other operating systems including Microsoft Windows.TM. series operating systems, Apple Macintosh 9 systems, Linux, Unix, SCO, BSD, FreeBSD, Microsoft Windows CE.TM., Microsoft Windows NT.TM., Microsoft Windows XP.TM., IBM AIX and OS/2.

[0041] With reference to FIG. 1, a method contained inside of a computer system is described as containing a file 1 that is being interrogated by a file comparator process 2 via an electronic link 6 to compute a hash signature and compare said signature to those contained in a database containing infected file signatures 4. The logical link 7 connecting the two processes and the file comparator 2 returning a result 3 of MATCH or NO MATCH.

[0042] With reference to FIG. 2, an end user computer 1 has a display 2 and a keyboard 3. The computer 1 additionally has a processing unit and a memory which provide (in functional terms) a graphical user interface layer 4 which provides data to the display 2 and receives data from the keyboard 3. The graphical user interface layer 4 is able to communicate with other computers via a network interface 5 and a network 6. The network is controlled by a network manager 7.

[0043] Beneath the graphical user interface layer 4, a number of user applications are run by the processing unit. In FIG. 2, only a single application 8 is illustrated and may be, for example, Microsoft Word.TM.. The application 8 communicates with a file system 9 which forms part of the Apple Macintosh OS X.TM. operating system and which is arranged to handle file access requests generated by the application 8. These access requests include file open requests, file save requests, file copy requests, etc. The lowermost layer of the operating system is the disk controller driver 10 which communicates with and controls the computer's hard disk drive 11. The disk controller driver 10 also forms part of the Apple Macintosh OS X.TM. operating system.

[0044] Located between the file system 9 and the disk controller driver 10 is a file system driver 12 which intercepts file system events generated by the file system 9. The role of the file system driver 12 is to co-ordinate virus screening and blocking operations for data being written to, or read from, the hard disk drive 11. A suitable file system driver 12 is, for example, the GATEKEEPER.TM. driver which forms part of the F-SECURE ANTI-VIRUS.TM. system available from Data Fellows Oy (Helsinki, Finland). In dependence upon certain screening operations to be described below, the file system driver 12 enables file system events to proceed normally or prevents file system events and issues appropriate alert messages to the file system 9.

[0045] The file system driver 12 is functionally connected to a virus print controller 13, such that file system events received by the file system driver 12 are relayed to the virus print controller 13. The virus print controller is associated with a database 14 which contain a set of "signatures" previously determined for respective infected files. For the purposes of this example, the signature used is a checksum derived using a suitable checksum calculation algorithm, such as the US Department of Defense Secure Hash Algorithm (SHA, SHA-1, SHA-224), MD5, MD2, or the older CRC 32 algorithm or other open source or proprietary algorithm capable of generating a hash signature value deemed acceptable to determine that one file is an identical copy of another file.

[0046] The database 14 contains a set of signatures derived for known viruses. Updates may be provided by way of floppy disks, CD, DVD, flash drive, FireWire, USB, or directly by downloading them from a remote server 17 connected to the Internet 18.

[0047] Only the network manager 7 and/or authorized computer administrator has the authority to modify this database14 using signatures specified by the anti-virus software provider.

[0048] Upon receipt of a file system event, the virus print controller 13 first analyses the file associated with the event (and which is intended to be written to the hard disk drive 11, read, copied, etc) to determine if the file matches that of a file identified to contain a virus.

[0049] The virus print controller 13 scans the database 14 to determine whether or not the corresponding signature is present in that database 14. If the signature is found there, the virus print controller 13 reports this to the file system driver 12. The file system driver 12 in turn causes the system event to be suspended and causes an alert to be displayed to the user that a known virus is present in the file. The file system driver 12 may also cause a report to be sent to the network manager 7 via the local network 6. The file system driver 12 quarantines the infected file on the hard disk drive 11.

[0050] The file scanning system described above is further illustrated by reference to the flow chart of FIG. 3.

[0051] It will be appreciated by the person of skill in the art that various modifications may be made to the embodiment described above without departing from the scope of the present invention. For example, the file system driver 12 may make use of further virus controllers including controllers arranged to screen files for viruses other than virus print identifiable. The file system driver 12 may also employ disinfection systems and data encryption systems.

[0052] It will also be appreciated that the file system driver 12 typically receives all file access traffic, and not only that relating to hard disk access. All access requests may be passed to the virus print controller 13 which may select only hard disk access requests for further processing or may also process other requests relating to, but not limited to, floppy disk data transfers, network data transfers, DVD, DVD-R, DVD-RW, CDROM, CD-RW, CD-R data transfers, USB, USB 2.0, FireWire, FireWire 2, and associated peripheral flash storage devices.

[0053] It will also be appreciated that the file system driver 12 and file system 9 along with applications 8 and GUI 4 can be those related to hand held, cell phone, PDA, digital camera, digital storage, or other devices containing a method to process electronic data as described above. It is also appreciated that hard disk drive 11 can be any electronic storage device such as flash, FireWire IEEE 1394, USB, USB 2.0, FireWire 2.0, and other electronic storage devices such as SD, MD, CF, etc. It is also appreciated that keyboard 3 can be any input device such as a cell phone keypad, microphone, or other electronic interface to a computer system or electronic device via wired or wireless connection.

[0054] With reference to FIG. 4, a method contained inside of a computer system is described as containing a file 1 that is being interrogated by a file comparator process 2 via an electronic link 6 to compute a hash signature and compare said signature to those contained in a database containing infected file signatures 4. The logical link 7 connecting the two processes and the file comparator 2 returning a result 3 of MATCH or NO MATCH.

[0055] In the case of data files in transit or when a complete file is not present or only pieces of a file are available. The file 1 is broken into several smaller blocks 8, 9, 10, and 11, for example, that are computed with unique hash signatures based on their size and location in the file as determined by the file comparator 2. The database 4 also contains hash signatures of these partial blocks wherein, for instance, the first block of data 8 may be a known and presetpercentage or piece of the file 1 under interrogation by start, end, and size of the partial file. The database 4 contains a complete hash for the file 1 as well as hash signatures for partial blocks 8, 9, 10, and 11, etc. The file comparator 2 interrogates the database to set starting and ending locations of known blocks of data to determine if itheata atis located the begging of a file 1 such as 8 or the end such as 11. Thus the comparator 2 can compute a hash and compare the hash for the partial file or block of data 8, 9, 10, or 11 f d match it with the appropriate signature location inside the database 4.